OBJECT ORIENTED PROGRAMMING WITH JAVA 8– LAB 10

1. Write a program to create two child threads using Thread class, one to compute the first 25 prime numbers and the other to compute the first 50 fibonacci numbers. After calculating 25 fibonacci numbers, make that thread to sleep and take up prime number computation.

Ans=

Code- Part 1,

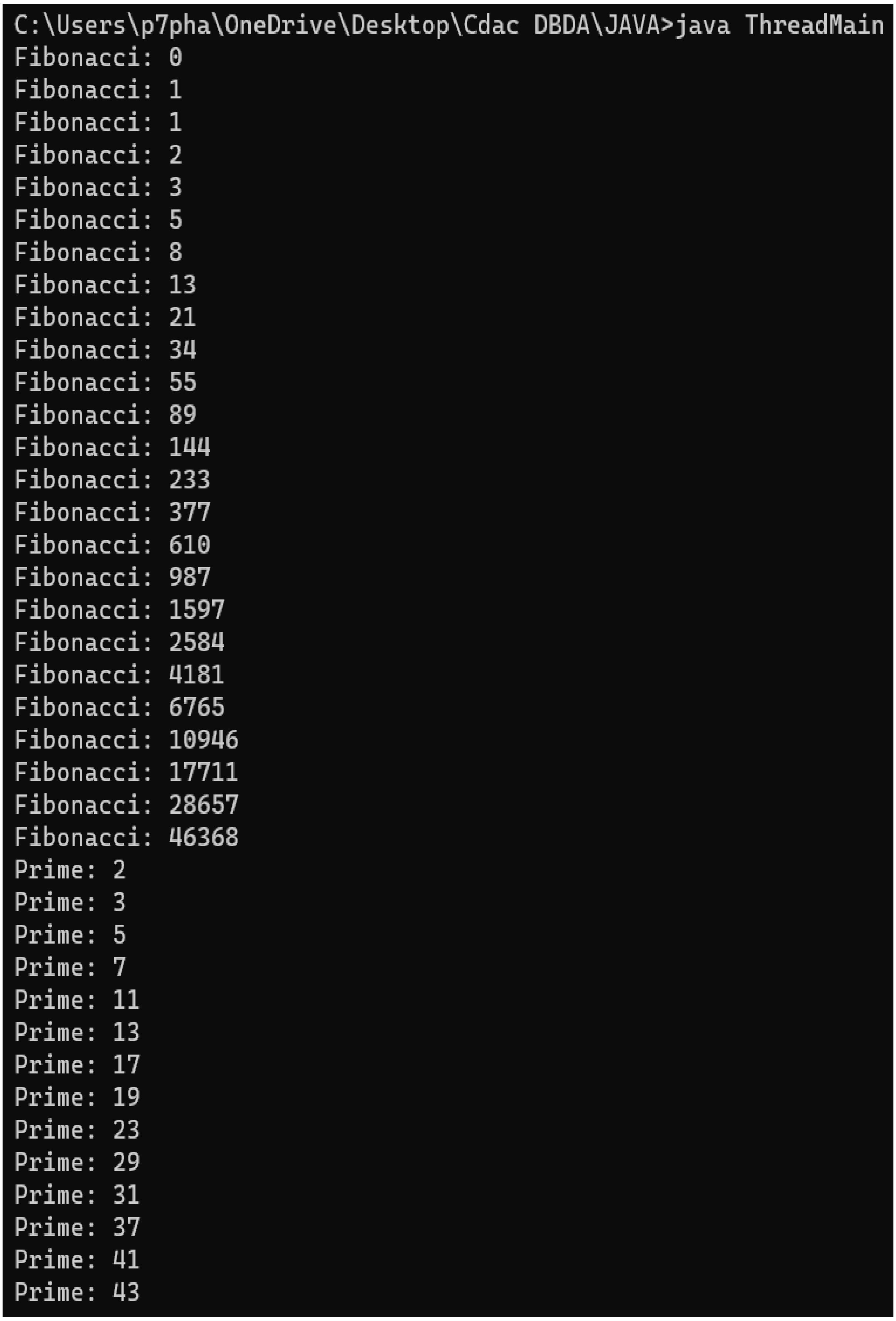
```
class PrimeThread extends Thread
{
        public void run()
        {
         int count = 0;
         int num = 2;

         while( count < 25)
                {
                 boolean Prime = true;
                 for(int i = 2; i <= Math.sqrt(num); i++)
                        {
                        if ( num % i == 0)
                                {
                                 Prime = false;
                                 break;
                                }
                        }
                        if (Prime)
                                {
                                 System.out.println("Prime: " + num);
                                 count++;
                                }
                 num++;
                }
        }
}
```

Part 2,

```
class FibonacciThread extends Thread
{
        public void run()
        {
         int count = 0;
         int a = 0;
         int b = 1;

         while ( count < 25)
                        {
                         System.out.println("Fibonacci: " +a);
                         int fibo = a + b;
                         a = b;
                         b = fibo;
                         count++;
                        }
        }
}
```

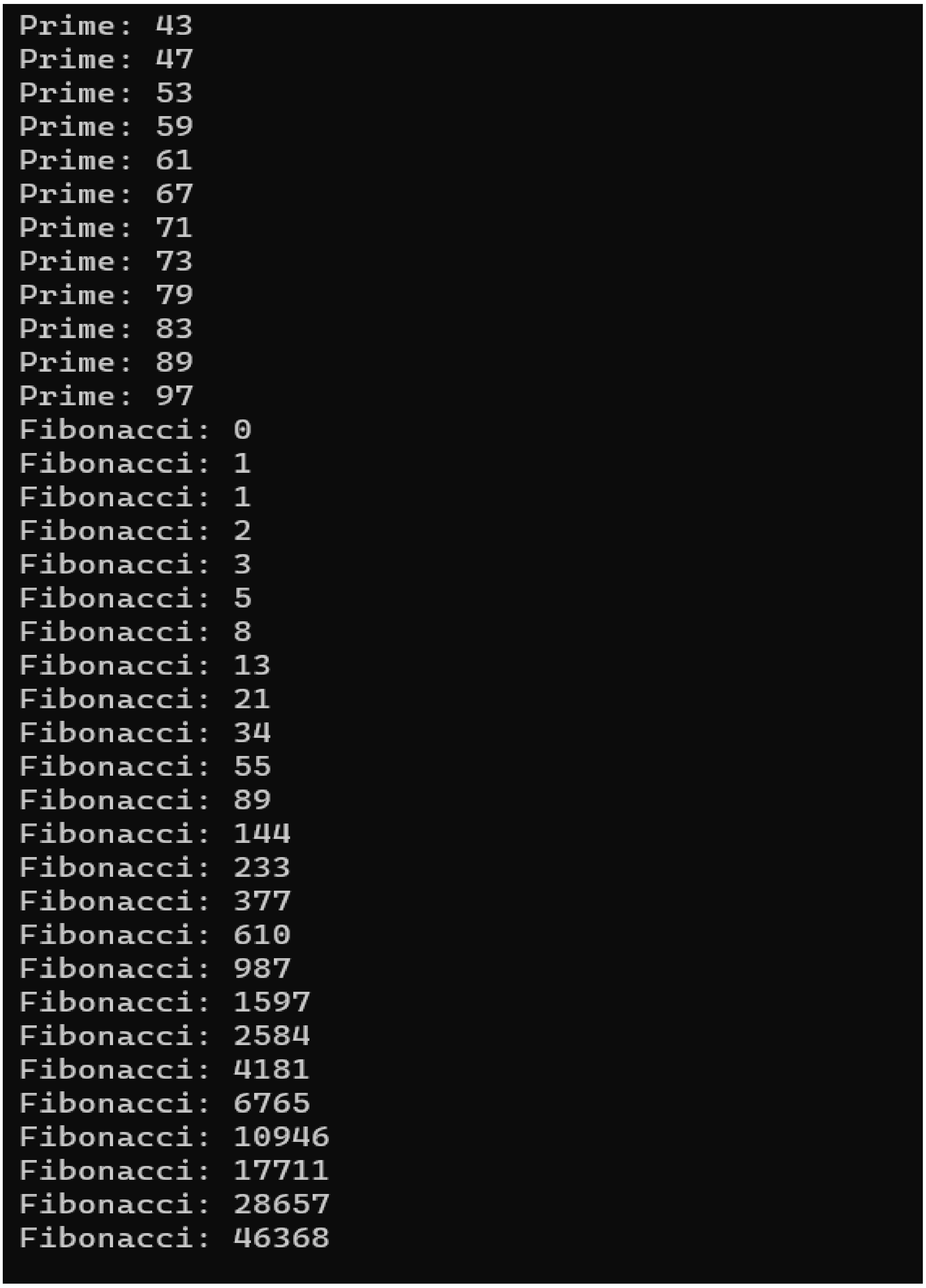
```
public class ThreadMain
{
        public static void main(String[] args)
        {
         FibonacciThread F = new FibonacciThread();
         PrimeThread P = new PrimeThread();

         F.start();
                        try
                           {
                            F.join();
                           }
                       catch(InterruptedException e)
                           {
                            System.out.println(e);
                           }
        P.start();
                        try
                          {
                           P.join();
                          }
                       catch (InterruptedException e)
                          {
                           System.out.println(e);
                          }


        int count = 25;
        int a = 0;
        int b = 1;

        while (count < 50)
                {
                 System.out.println("Fibonacci: " + a);
                 int fibo = a + b;
                 a = b;
                 b = fibo;
                 count++;
                }
        }
}
```

Execution- Part 1,



```
C:\Users\p7pha\OneDrive\Desktop\Cdac DBDA\JAVA>java ThreadMain
Fibonacci: 0
Fibonacci: 1
Fibonacci: 1
Fibonacci: 2
Fibonacci: 3
Fibonacci: 5
Fibonacci: 8
Fibonacci: 13
Fibonacci: 21
Fibonacci: 34
Fibonacci: 55
Fibonacci: 89
Fibonacci: 144
Fibonacci: 233
Fibonacci: 377
Fibonacci: 610
Fibonacci: 987
Fibonacci: 1597
Fibonacci: 2584
Fibonacci: 4181
Fibonacci: 6765
Fibonacci: 10946
Fibonacci: 17711
Fibonacci: 28657
Fibonacci: 46368
Prime: 2
Prime: 3
Prime: 5
Prime: 7
Prime: 11
Prime: 13
Prime: 17
Prime: 19
Prime: 23
Prime: 29
Prime: 31
Prime: 37
Prime: 41
Prime: 43
```

Part 2,



```
Prime: 43
Prime: 47
Prime: 53
Prime: 59
Prime: 61
Prime: 67
Prime: 71
Prime: 73
Prime: 79
Prime: 83
Prime: 89
Prime: 97
Fibonacci: 0
Fibonacci: 1
Fibonacci: 1
Fibonacci: 2
Fibonacci: 3
Fibonacci: 5
Fibonacci: 8
Fibonacci: 13
Fibonacci: 21
Fibonacci: 34
Fibonacci: 55
Fibonacci: 89
Fibonacci: 144
Fibonacci: 233
Fibonacci: 377
Fibonacci: 610
Fibonacci: 987
Fibonacci: 1597
Fibonacci: 2584
Fibonacci: 4181
Fibonacci: 6765
Fibonacci: 10946
Fibonacci: 17711
Fibonacci: 28657
Fibonacci: 46368
```

2. Write a Java program to create two child threads using Runnable interface, one to find the sum of natural numbers from 1 to n and the other to display the factorial of the numbers upto n.

Ans=

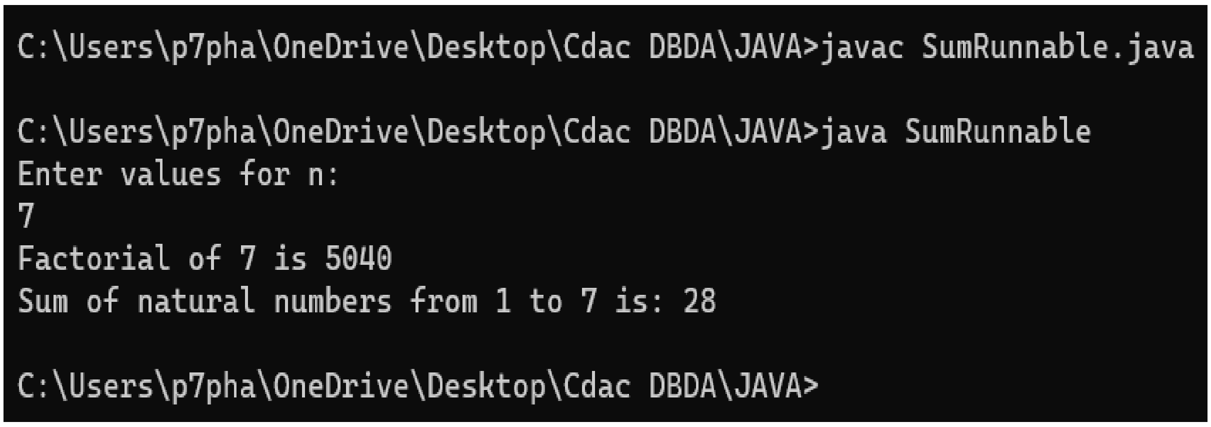
Code- Part 1,

```
import java.util.Scanner;

class SumThread implements Runnable
{
        int n;

        public SumThread(int n)
        {
        this.n = n;
        }

        public void run()
        {
        int sum = 0;
        for (int i = 1; i <= n; i++)
                {
                 sum += i;
                }
        System.out.println("Sum of natural numbers from 1 to " + n + " is: " + sum);
        }
}
```

Part 2,

```
class FactorialThread implements Runnable
{
        int n;

        public FactorialThread(int n)
        {
         this.n = n;
        }

        public void run()
        {
        int factorial = 1;
        for (int i = 1; i <= n; i++)
                {
                 factorial *= i;
                }
        System.out.println("Factorial of " + n + " is " + factorial);
        }
}

public class SumRunnable
{
        public static void main(String[] args)
        {
        Scanner P = new Scanner(System.in);

        System.out.println("Enter values for n: ");
        int n = P.nextInt();

        Thread sumThread = new Thread(new SumThread(n));
        Thread factorialThread = new Thread(new FactorialThread(n));

        sumThread.start();
        factorialThread.start();
        }
}
```

Execution-

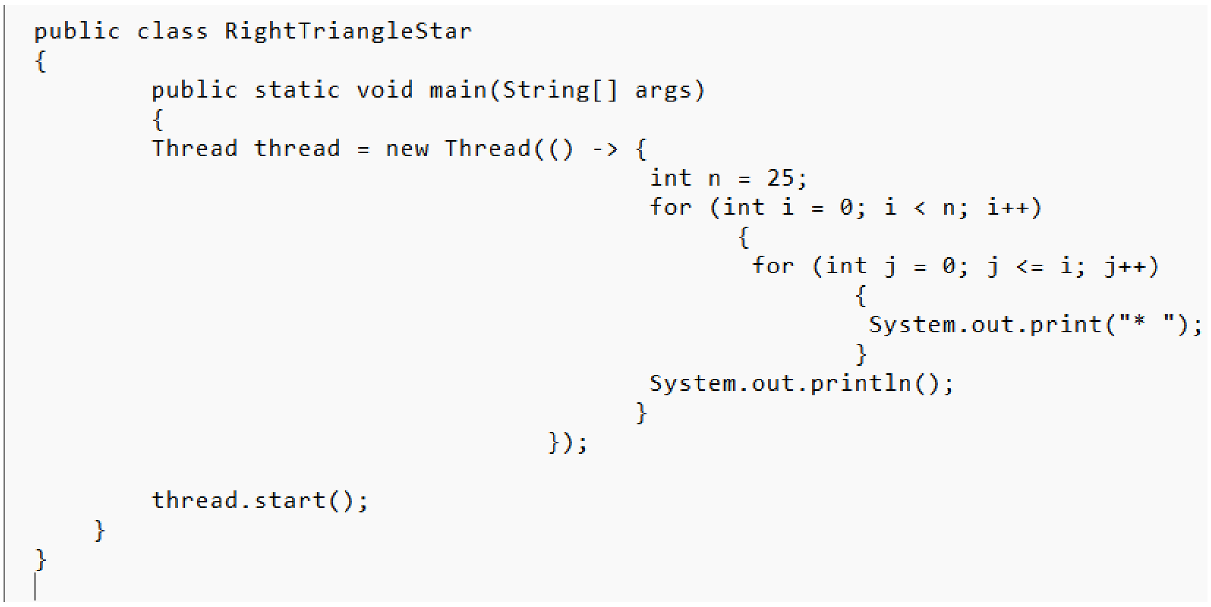


```
C:\Users\p7pha\OneDrive\Desktop\Cdac DBDA\JAVA>javac SumRunnable.java

C:\Users\p7pha\OneDrive\Desktop\Cdac DBDA\JAVA>java SumRunnable
Enter values for n:
7
Factorial of 7 is 5040
Sum of natural numbers from 1 to 7 is: 28

C:\Users\p7pha\OneDrive\Desktop\Cdac DBDA\JAVA>
```

3. Create a child thread to print Right Triangle star pattern using Lambda expression.

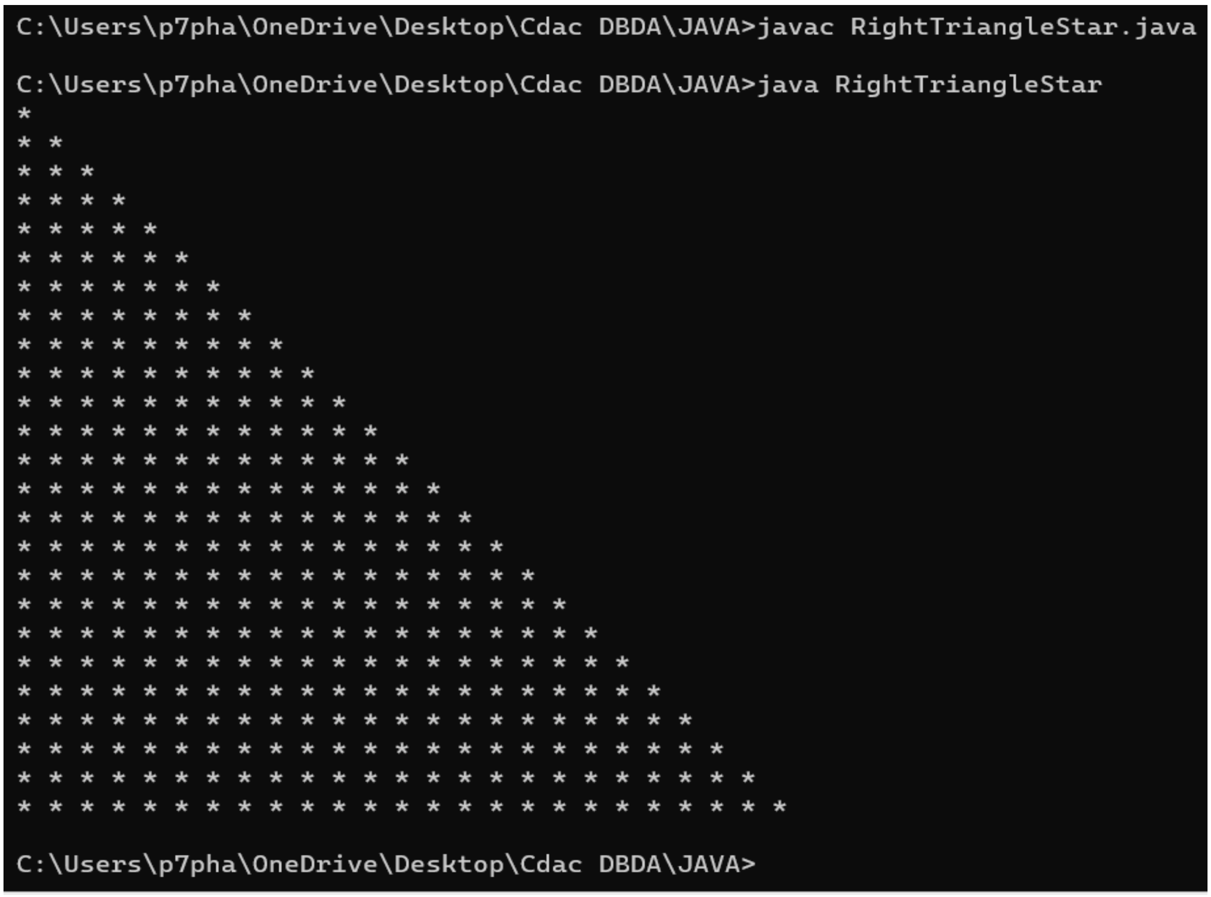
Ans=

Code-



```
public class RightTriangleStar
{
        public static void main(String[] args)
        {
        Thread thread = new Thread(() -> {
                                          int n = 25;
                                          for (int i = 0; i < n; i++)
                                                {
                                                 for (int j = 0; j <= i; j++)
                                                        {
                                                         System.out.print("* ");
                                                        }
                                          System.out.println();
                                         }
                                   });

        thread.start();
    }
}
```

Execution-



```
C:\Users\p7pha\OneDrive\Desktop\Cdac DBDA\JAVA>javac RightTriangleStar.java

C:\Users\p7pha\OneDrive\Desktop\Cdac DBDA\JAVA>java RightTriangleStar
*
* *
* * *
* * * *
* * * * *
* * * * * *
* * * * * * *
* * * * * * * *
* * * * * * * * *
* * * * * * * * * *
* * * * * * * * * * *
* * * * * * * * * * * *
* * * * * * * * * * * * *
* * * * * * * * * * * * * *
* * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * *
* * * * * * * * * * * * * * * * * * * * * * * * *

C:\Users\p7pha\OneDrive\Desktop\Cdac DBDA\JAVA>
```

4. Create a BankAccount class with data members accno, balance and methods deposit and withdraw. Create an object for it which is a joint account. Two threads are using the same account. One thread is trying to deposit an amount to that account and second thread trying to withdraw an amount from it after checking the minimum balance. Implement the program using synchronization.

Ans=

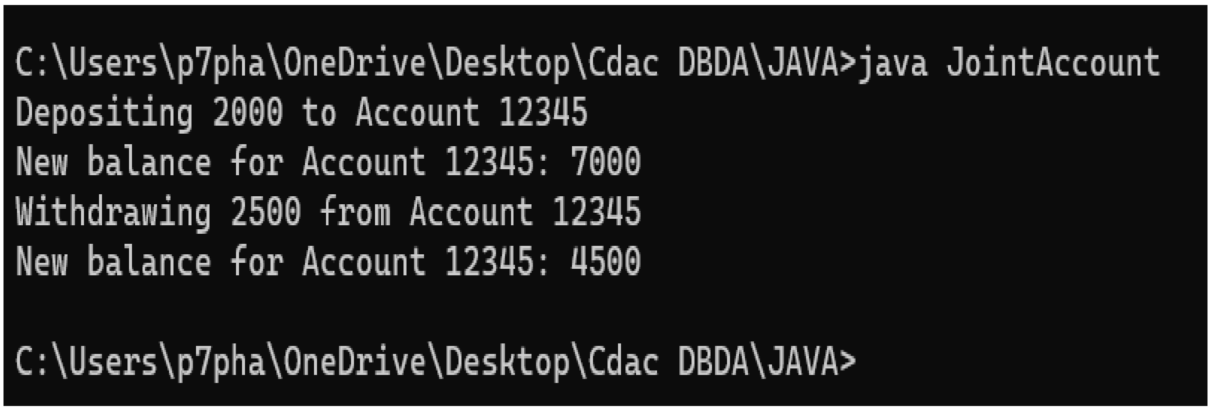
Code- Part 1,

```
class BankAccount
{
        int accno;
        int balance;

        public BankAccount(int accno, int initialBalance)
        {
         this.accno = accno;
         this.balance = initialBalance;
        }

        public synchronized void deposit(int amt)
        {
         System.out.println("Depositing " + amt + " to Account " + accno);
         balance += amt;
         System.out.println("New balance for Account " + accno + ": " + balance);
        }

        public synchronized void withdraw(int amt)
        {
         if (balance - amt >= 1000.0)
                {
                 System.out.println("Withdrawing " + amt + " from Account " + accno);
                 balance -= amt;
                 System.out.println("New balance for Account " + accno + ": " + balance);
                }
            else
                {
                 System.out.println("Withdrawal denied for Account " + accno + ". Insufficient balance.");
                }
        }
}
```

Part 2,

```
public class JointAccount
{
         public static void main(String[] args)
         {
          BankAccount jointAcco = new BankAccount(12345, 5000);

          Thread depositThread = new Thread(() -> {
                                                  jointAcco.deposit(2000);
                                                });

          Thread withdrawThread = new Thread(() -> {
                                                   jointAcco.withdraw(2500);
                                                  });

         depositThread.start();
         withdrawThread.start();
        }
}
```

Execution-



```
C:\Users\p7pha\OneDrive\Desktop\Cdac DBDA\JAVA>java JointAccount
Depositing 2000 to Account 12345
New balance for Account 12345: 7000
Withdrawing 2500 from Account 12345
New balance for Account 12345: 4500

C:\Users\p7pha\OneDrive\Desktop\Cdac DBDA\JAVA>
```